Sam Bartlett

sbartlett30@gatech.edu

HelperBot : A Prototype System for Reducing Cognitive Load

Introduction

This project is a continuation of research into finding ways to reduce the high attrition rate of online and non-traditional college students by reducing their cognitive load. Cognitive Load Theory (CLT), states that people have a limited working memory, for processing new information and ideas, that can store about seven elements at any given time. This information is then later transferred to long term memory (Kirschner P.A., 2012). Not only does CLT plays an important role in student learning, but also their ability to plan and organize to keep up to date with assignments, tests, and projects. For non-traditional students, whose working memory may be close to full with concerns about work and family concerns, thoughts about getting homework completed may get pushed out before it is transferred. The concept of this study is to help reduce a student's overall cognitive load using reminders for school related matters, which should aid in completing coursework, improve student performance, and thereby reduce the attrition rate.

Prior to HelperBot, I designed and tested two other prototype systems, a mobile app/website, and email reminders. What follows is a brief overview of each.

Mobile App and Website

For the first attempt, I took a more holistic approach and built a mobile app and website for an online history class that I taught. The app allowed students to see at-a-glance the assignments due each week, take notes as needed for the class, and add reminders to their phone's calendar with a single click. They also had a checklist they could use by checking off the assignments as they completed them.

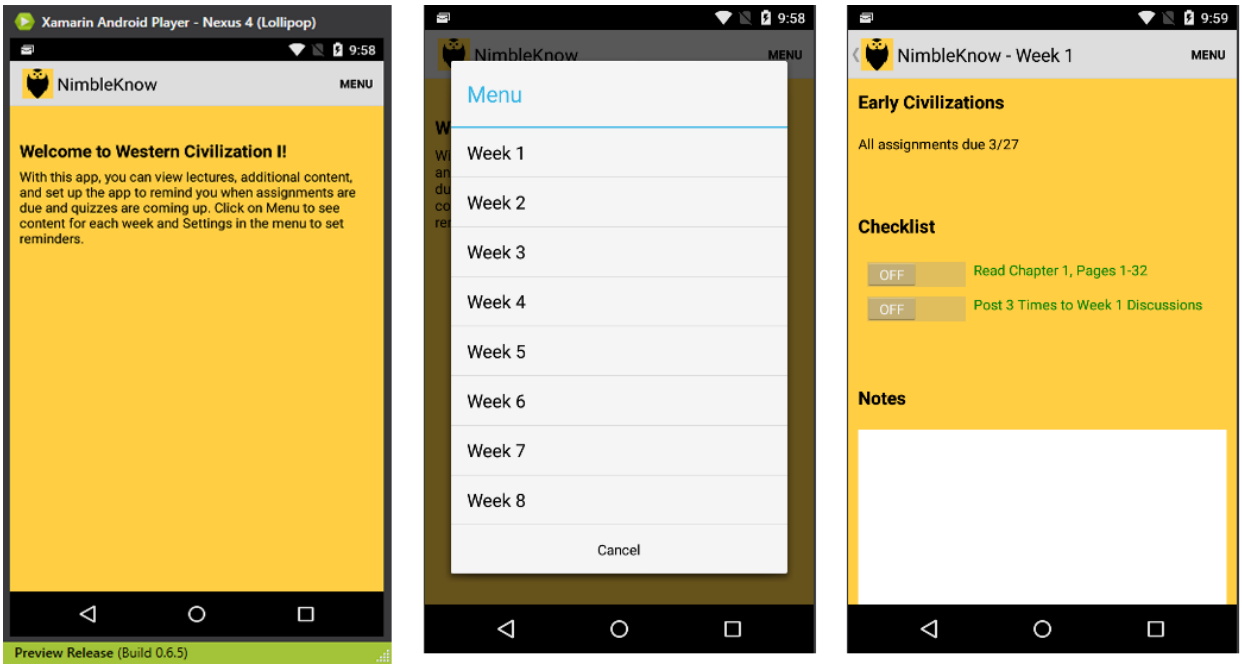

Figure 1, the NimbleKnow mobile app

Students could save notes and check marks on the app. The website was built as a backup for students who did not have smartphone or who chose not to install the app. The website worked in a similar way, except there were no save features. Students were recruited from a section of the course. Students were free to either use the app on iOS or Android, or the mobile website. Students were requested to use the app or website and then take a survey.

The survey consisted of sixteen questions in three sections. The first section asked respondents about their college experience; the second section were about issues that could potentially impact a student's cognitive load; and the third and last section was to discover if students had tried the app or the website. While the sample size (fourteen) was too small to be scientific, it had some interesting anecdotal data. Most survey takers responded that they faced daily challenges keeping up with assignments due their work and family responsibilities, or in other words, increased cognitive load. For the next two projects, I chose only to focus on reducing cognitive load.

Email Reminders

For the second project, I wanted to build on the first project, but have a larger sample size that could be more statistically significant. Also, the project would only focus on helping students keep track of their assignments to reduce their cognitive load. I was able to scale the project size up for multiple instructors and courses by choosing to work with the Cousera MOOC platform. I first planned on tracking students on attendance, participation in discussions, homework completion/grades, test completion/grades, final grades, and if they completed the course or dropped prior to completion. I also wanted to do two runs of the experiment, given that Coursera semesters are short.

The first run would use only positive reminders, such as, “keep up the good work!”, and in the second round, do more varied messaging between positive, neutral, and negative messaging. There would also be a website for students to opt-in to the study and select the courses they were presently enrolled in. Then using Coursera’s API, the web site would pull in coursework data for a class and send students reminders via email at varying frequencies using Constant Contact’s API. However, due to unforeseen complications, the study did not go as planned, and work on the platform was postponed.

I asked professors at Georgia Tech who teach classes with the Coursera platform to participate in the study and requested that they help recruit their students for the experiment. For the first round, one instructor and fifteen of out of ninety-seven students participated. Students opted in through an email opt-in form created on Constant Contact’s website. The original concept was to use their name and email address from the opt-in form and tie it back to a data export from Coursera once the course ended so I could see if the reminders had a noticeable impact. This first class only required students to do weekly readings and quizzes. So instead of sending reminders per assignment, I opted to send email reminders twice a week, one five days before the quiz and readings were due, one email the day before

they were due. Coursera also sent out a weekly reminder, two days prior to the due date. The average open rate for the emails was 25%. At the end of the first run of the experiment, I sent out the survey, hosted on SurveyPlanet. Only two students responded.

For the second run of the experiment, another instructor offered to participate who taught three class. Seventy-three students opted in. However, her courses were for students whose primary language was not English, which added an unexpected variable. The information on who was taking what course and how many students were in these classes was not available. I attempted to contact students directly and find out what courses they were in, but only two responded. Despite the lack of metrics, I persisted. I chose to modify the experiment slightly, sending only one reminder a week. Open rates on the email reminders averaged 21%.

During the second run I received the Coursera class data from the first run. After spending time sifting through the data, it was apparent that there was no way to tie the student names and email to the data export, and therefore I could not tell if the reminders were having an impact. I never received any responses from my contacts at Coursera to my request for this information. Consequently, the final survey results offer only a window into the experiment's results, but not a complete overview.

I developed the survey using Survey Planet's online platform. The survey consisted of nineteen questions divided into three sections. The first section, which had three questions, asked respondents about their college experience, the second section of nine questions were about issues that could provide some insight into a student's personal cognitive load. The last section were questions related to students' experience with the reminders from both my experiment and Coursera. Fifteen students responded to the survey. Overall the survey results and participation rates demonstrated some interest in using reminders to help reduce cognitive load, but without correlated evidence in terms of student course completion and academic performance, this information was anecdotal at best. For the last and

final project, I focused on building a prototype chatbot that could interact with students as well do push reminders.

HelperBot

The idea for HelperBot is to combine the push functionality of a mobile app for reminders with the ease of use of a conversation bot. As with the second project, the focus is on reducing students cognitive load through the use of reminders. The original scope of the project was to build the bot, use it for a class hosted on edX's MOOC platform, and then review grades and attrition rates for effectiveness. HelperBot would be able to interact with the students, giving them what was due for the present week, allow them to register for push notifications, and answer basic questions students might have about the course. However, complications with data and the bot channel utilized led to delays, changing the scope and scale from an experiment to a prototype build.

HelperBot's starting architecture was to use edX's API to pull in grades and student data, then use an Admin Console Website to set parameters for frequency reminders, view analytics of student interaction with the bot, and add to the bot's knowledgebase for frequently asked questions.

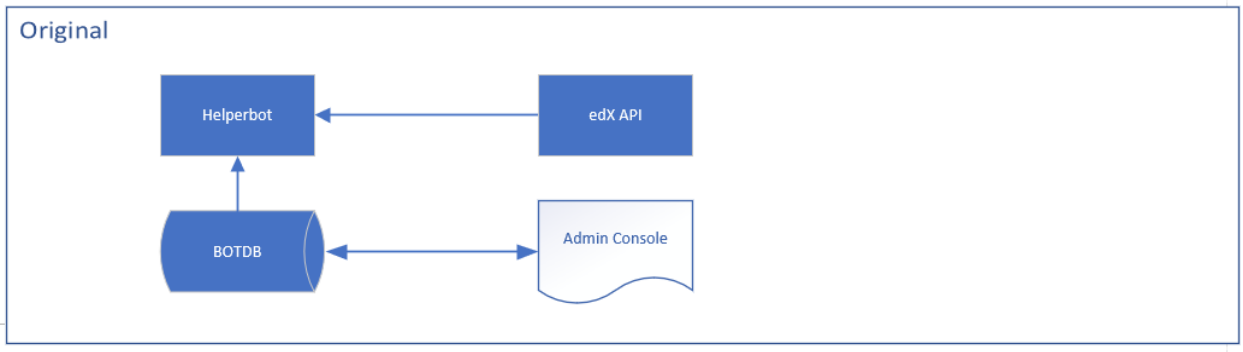

Figure 2, the original HelperBot architecture

This bot is built on Microsoft's Bot Framework (version 3)[1], written in C#, and hosted on the Azure cloud platform, using its Bot Service and SQL Data Storage.[2] I used the Echo Bot template as the foundation for HelperBot.

After further research I discovered that edX's API was still in alpha and did not provide a way to pull student or grading data out dynamically. Instead, I had to manually request comma delimited files in a CSV format for both student data and grading information. I used SQL Server Management Studio to import the first set of data. I then added a CSV import process to the architecture where the Admin Console would also then upload the CSV file. Later, I discovered additional challenges with working with the data.

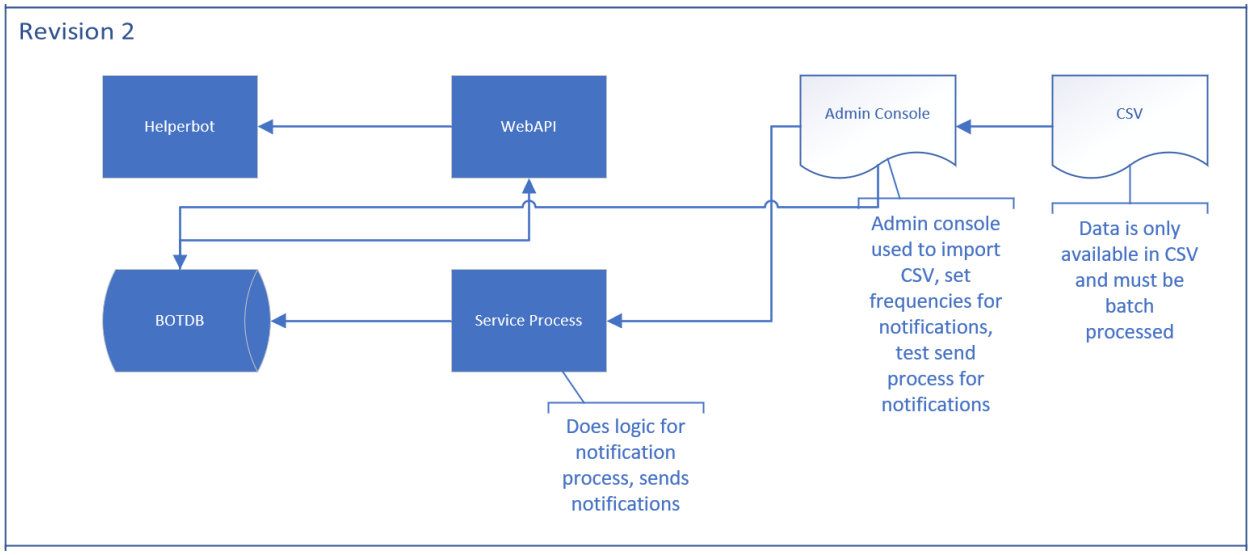

Figure 3, Updated Architecture for HelperBot

The schedule of assignments with due dates was not available as a single CSV so I created the table and data manually. Now students could ask HelperBot what was due for that current week.

[1] https://dev.botframework.com/, (Microsoft, n.d.).

[2] https://docs.microsoft.com/en-us/azure/bot-service/?view=azure-bot-service-3.0, (Microsoft, n.d.).

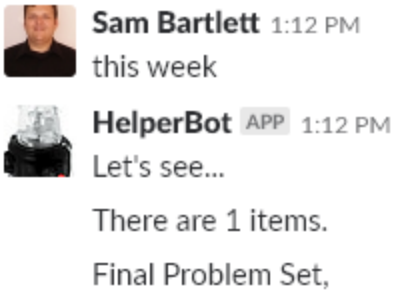

Figure 4, HelperBot Listing the Current Week's Assignments

I then turned to focus on the building the notification process that HelperBot would use to push notifications. For the course I planned to work on, Slack was used for a communications platform. Fortunately, Azure Bot Framework allows you to connect a bot to any number of communication channels, so connecting HelperBot to Slack was simple. There was not an obvious way to push notifications to students. First, I had no way to tie a student back to the gradebook using Slack handles, which is where the registration dialog for HelperBot came in. Students type in the phrase “add me” to get added for push notifications.

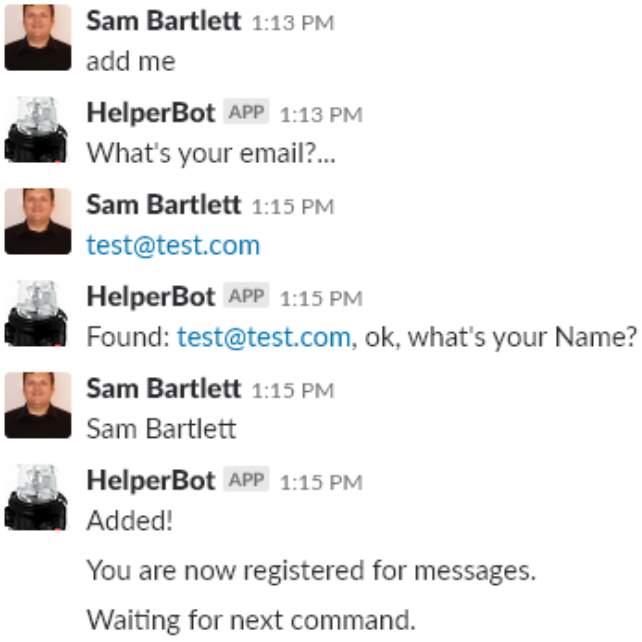

Figure 5, HelperBot's Registration Dialog

I first attempted to use webhooks to test push notifications to students on Slack, using Piotr Ganekiewicz's instructions as a guide (Gankiewicz, n.d.). This would only work if I manually added a webhook to HelperBot, for each individual student. There was no way to programmatically add webhooks to HelperBot. I searched for another solution. I came across an article on StackOverflow that pointed me in the right direction ("VzLOM", n.d.) . initial attempts were not successful, as I ran into Slack-related errors. After about a week of troubleshooting, I got the push notification tests to work.

Throughout this period, I continued to refine the architecture of the prototype. I would need a way to have the system that supported HelperBot send the reminders at frequencies specified in the database by the administrator of the system. I considered using a separate service process that would use Azure's WebJobs, which is similar to a Windows Service, but came across a web-based solution, Hangfire, (Hangfire, n.d.) that I could integrate into the same base as HelperBot, eliminating a separate service.

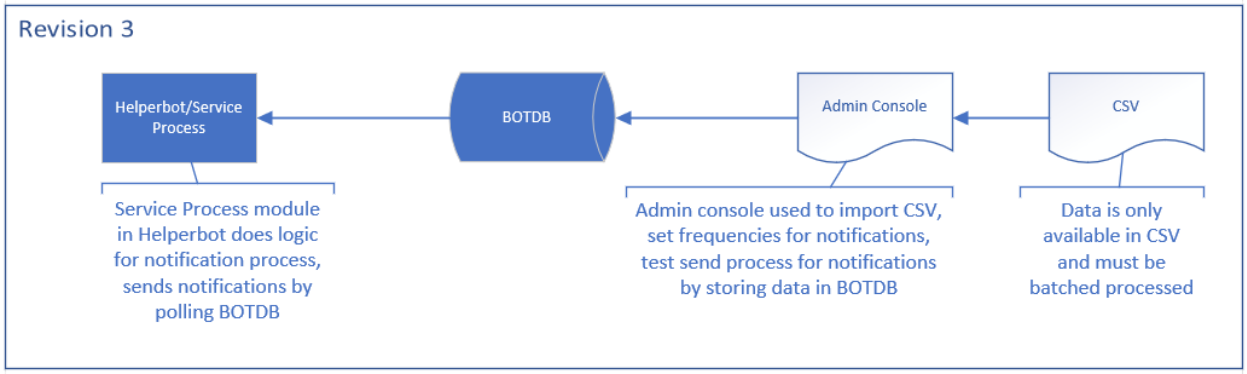

Figure 6, Third Revision of HelperBot's System Architecture

For HelperBot, it would send push messages based on the frequencies set by the system administrator for two events- one for when a student's grade for an assignment was below 60% and

another where a student had not turned in an assignment.

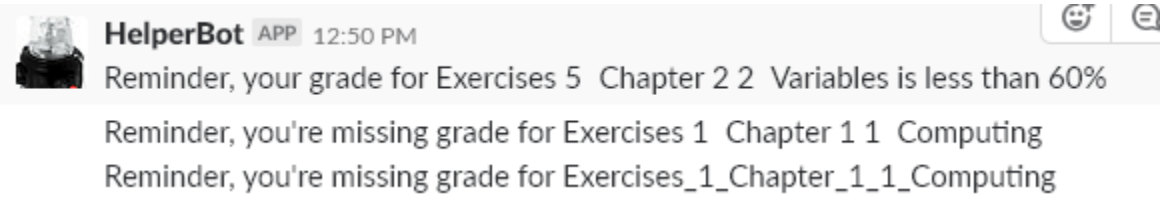

Figure 7, an Example of Push Notifications to a Student from HelperBot

At this point, I chose to integrate the administrative functions into HelperBot with a password protected dialog tree. This way I could eliminate creating a separate web site that would need to be running and maintained outside of HelperBot, and the administrator would get to use a bot instead of a static website. The administrative menu functions include:

- View analytics – see recent student activity
- View prefs – see preferences set for the two reminder types
- Set prefs – set preferences for the reminder interval for the two reminder types
- Mastercontrol – the entry point into the admin dialog, password protected

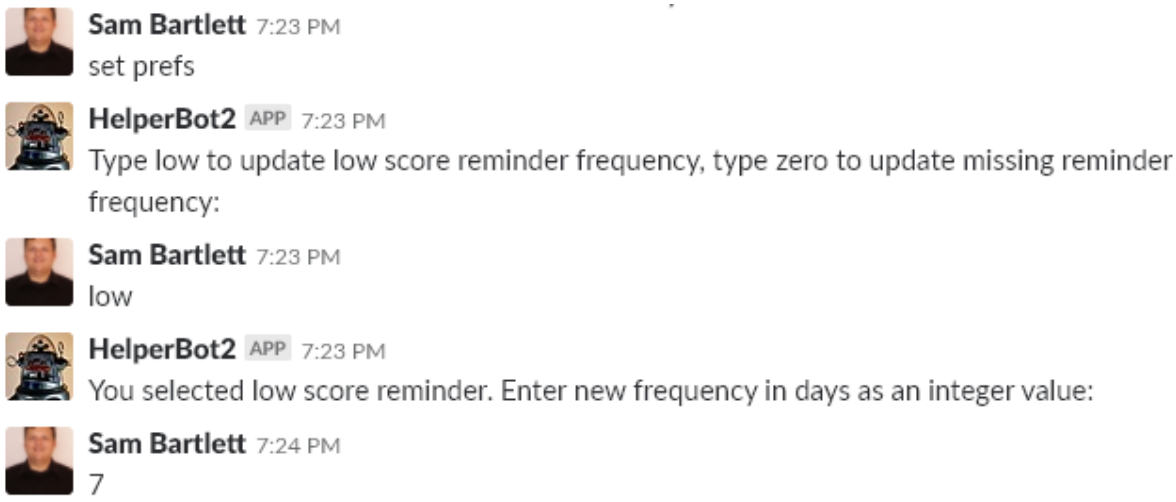

Figure 8, Demonstration of Admin Functions in HelperBot

Continued updates to HelperBot's code ran into a snag inside of Slack. At one point, HelperBot stopped responding. After numerous communications with the support team at Slack failed to find an issue, I resolved to completely delete and recreate HelperBot inside of Slack. I had to rename HelperBot to HelperBot2 for the recreation process to work.[3] The final piece of the puzzle left to solve was the format of the data that I was downloading from edX.

While there is presently no way of getting around the manual download of the data, the other issue with it was the way the data is presented in the CSV. The problem grade report data proved to be unfeasible to incorporate at this state. It contained over 1,200 columns even after some clean up, and it was too large to import into the database. For a future version of HelperBot, a no-SQL database might be worth testing. The gradebook data was less daunting. However, the way it was organized was not suitable for what HelperBot needed.

HelperBot needed the data organized so that each assignment was tied to specific date, so it would send reminders about the correct assignments for a given timeframe. I also needed the data structured so that I would have the grades per student, per assignment, per date. This led to the final addition to the architecture, a Windows console app to transpose the data and insert it into the database. The process took about ten minutes in its initial run and created a many to many table with over 10,000 rows. The import tool would have to be run weekly by an administrator for the data to stay current. Once edX's API is further along, it may be possible to replace the database import process with a direct API call. With all the components in the prototype system complete, it is ready for testing and further refinement.

[3] Hence the difference in names in figure 8. I changed the picture of the bot for purely aesthetic reasons.

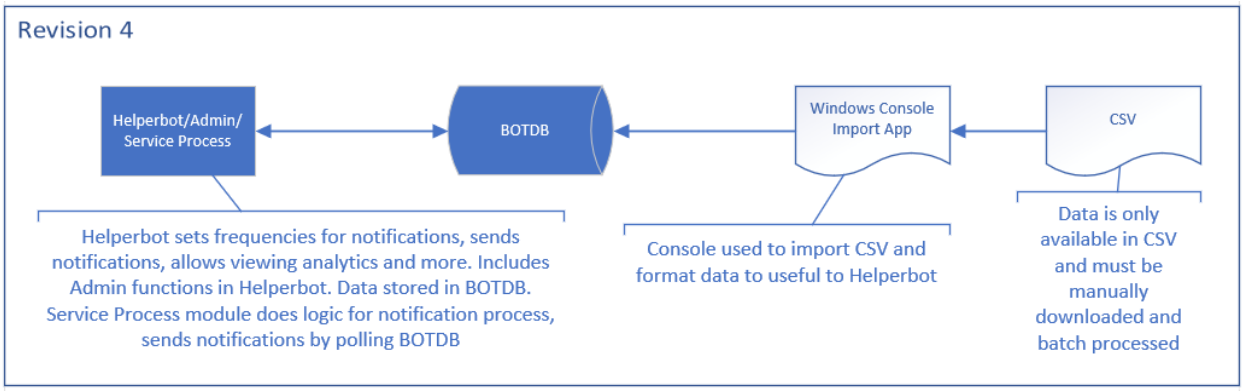

Figure 9, the Final Design of the HelperBot Prototype

Conclusion

The creation of a working HelperBot prototype was more challenging and took far longer to complete than expected, given the difficulties with data and Slack's infrastructure. With the initial hurdles out of the way, there are many ways to add more functionality to the bot, for both students and course administrators, including adding natural language processing, a knowledgebase of frequently asked questions, and more flexibility to add more types of reminders with their own frequencies. The next stage would be to run a test with the bot and see if it has an impact on reducing students' cognitive load, thereby improving their performance and reducing attrition, which was the original goal of this study.

References

"VzLOM". (n.d.). *Is there a way to start formflow dialog on ConversationUpdate event?* Retrieved from StackOverflow: https://stackoverflow.com/questions/44353520/is-there-a-way-to-start-formflow-dialog-on-conversationupdate-event

Aleahmad, T. (2012, May 7). Improving Students' Study Practices Through the Principled Design of Research Probes. Retrieved from http://repository.cmu.edu/dissertations/129/

Angelino, L. M., Williams, F. K., & Natvig, D. (2007). Strategies to Engage Online Students and Reduce Attrition Rates. *Journal of Educators Online, 4*, 1-14.

Benda, K., Bruckman, A., & Guzdial, M. (2012, November). When Life and Learning Do Not Fit: Challenges of Workload and Communciation in Introductory Computer Science Online. *ACM Transactions on Computing Education, 12*(4), 3-38.

Bernard, R. M., Abrami, P. C., Lou, Y., Borokhovsk, E., Wade, A., Wozney, L., . . . Huang, B. (2004). How Does Distance Education Compare to Classroom Instruction? A Meta-Analysis of the Empirical Literature. *Review of Educational Research, 73*(Fall), 379-439.

Bradford, G. R. (2011). A Relationship Study of Sutdent Satisfaction with Learning Online and Cognitive Load: Initial Results. *The Internet and Higher Education*, 217-226.

Brawer, F. (1196). *Retention-Attrition in the Nineties, ERIC Digest.* Los Angeles: ERIC Clearinghouse for Community Colleges.

Brawer, F. B. (1996, April). Retention-Attrition in the Nineties. ERIC Digest. Retrieved from http://files.eric.ed.gov/fulltext/ED393510.pdf

Cabrera, A. F., Castaneda, M. B., Nora, A., & Hengstler. (1992). The Convergence between Two Theories of College Persistence. *Journal of Higher Education, 63*, 143-164.

D. Song, E. Y. (2017). Interacting with a conversational agent system for educational purposes in online courses. *10th International Conference on Human System Interactions (HSI)* (pp. 78-82). Ulsan, Korea: IEEE .

Daempfle, P. A. (n.d.). An Analysis of the High Attrition Rates among First Year College Science, Math and Engineering Majors. Retrieved from http://files.eric.ed.gov/fulltext/ED465347.pdf

Davidson, W. B., Beck, H. P., & Milligan, M. (2009). The College Persistence Questionnaire: Develeopment and Validation of an Instrument That Predicts Student Attrition. *Jounral of College Student Development, 50*(July/August), 373-390.

De Bruyckere, P., Kirschner, P. A., & Hulshof, C. D. (2015). *Urban Myths About Learning and Education.* New York: Elsevier.

DeBerard, M. S., Spielmans, G. I., & Julka, D. C. (2004). Predictors Of Academic Achievement And Retention Amongcollege Freshmen: A Longitudinal Study. *College Student Journal, 38*(1), 66-80.

Eaton, S. B., & Bean, J. P. (1995). An Approach/Avoidance Behavioral Model of College Student Attrition. *Research in Higher Education, 36*, 617-645.

Foster, G. (2010). Teacher Effects on Student Attrition and Performance in Mass-Market Tertiary Education. *Higher Education: The International Journal of Higher Education and Educational Planning, 60*, 301-319.

Gankiewicz, P. (n.d.). *Effortless C# integration with Slack*. Retrieved from Piotr Gankiewicz: https://piotrgankiewicz.com/2016/06/10/effortless-csharp-integration-with-slack/

Hangfire. (n.d.). Retrieved from Hangfire: https://www.hangfire.io/

Hart, C. (2012). Factors Associated With Student Persistence in an Online Program of Study: A Review of the Literature. *Journal of Interactive Online Learning*, 19-39.

Heller, B. &. (2007). Conversational Agents and Learning Outcomes: An Experimental Investigation. In C. Montgomerie & J. Seale (Eds.). *Proceedings of ED-MEDIA 2007--World Conference on Educational Multimedia, Hypermedia & Telecommunications* (pp. 945-950). Vancouver, Canada: Association for the Advancement of Computing in Education (AACE).

Heller, B. P. (2005). Freudbot: An Investigation of Chatbot Technology in Distance Education. In P. Kommers & G. Richards (Eds.). *Proceedings of ED-MEDIA 2005--World Conference on Educational Multimedia, Hypermedia & Telecommunications* (pp. 3913-3918). Montreal, Canada: Association for the Advancement of Computing in Education (AACE).

Johnson, S. D., & Aragon, S. R. (2003). An Instructional Strategy Framework for Online Learning Environments. *New Directions for Adult and Continuing Education*(100), 31-43.

Karp, M. M., Hughes, K. L., & O'Gara, L. (n.d.). An Exploration of Tinto's Integration Framework for Community College Students. CCRC Working Paper No. 12. Retrieved from http://files.eric.ed.gov/fulltext/ED501335.pdf

Kerlyl A., H. P. (2007). *Bringing Chatbots into education: Towards Natural Language Negotiation of Open Learner Models. In: Ellis R., Allen T., Tuson A. (eds) .* London : Springer.

Kerry A., E. R. (2009). *Conversational Agents in E-Learning. In: Allen T., Ellis R., Petridis M. (eds) Applications and Innovations in Intelligent Systems XVI. SGAI 2008.* London: Springer.

Kirschner, P. A. (2012). Cognitive Load Theory: Implications of Cognitive Load Theory on the Design of Learning. *Learning and Instruction*(12), 1-10.

Kirschner, P. A. (2013). Do Learners Really Know Best? Urban Legends in Education. *Educational Psychologist*, 169-183.

Lotkowski, V. A., Robbins, S. B., & Noeth, R. J. (2004). The Role of Academic and Non-Academic Factors in Improving College Retention. *ACCT Policy Report*. Retrieved from http://files.eric.ed.gov/fulltext/ED485476.pdf

Metz, G. W. (2004). Challenge and Changes to Tinto's Persistence Theory: A Historical Review. *Journal of College Student Retention, 6*(2), 191-207.

Microsoft. (n.d.). Retrieved from Microsoft Bot Framework: https://dev.botframework.com/

Microsoft. (n.d.). *Create a bot with Azure Bot Service*. Retrieved from Bot Service - Microsoft Docs: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-quickstart?view=azure-bot-service-3.0

Morrison, B. B. (2013). Using Cognitive Load Theory to Improve the Effeciency of Learning to Program. *ICER '13* (pp. 183-184). New York: ACM.

Morrison, B. B. (2015, October). Replicating Experiments from Educational Psychology to Develop Insights Into Computing Education: Cognitive Load as a Significant Problem in Learning Programming.

Ozga, J., & Sukhnandan, L. (1998). Undergraduate Non-Completion: Developing an Explanatory Model. *Higher Education Quarterly, 52*(3), 316-333.

Pascarella, E. T., & Terenzini, P. T. (1979). Student-Faculty Informal Contact and College Persistence: A Further Investigation. *The Journal of Educational Research, 72*(Mar. - Apr.), 214-218.

Rovai, A. (2003). In Search of Higher Persistence Rate in Distance Education Online Programs. *Internet and Higher Education, 6*, 1-16.

Terenzini, P. T., & Pascarella, E. T. (1980). Toward the Validation of Tinto's Model of College Student Attrition: A Review of Recent Studies. *Research in Higher Education, 12*, 271-282.

The New York Times Editorial Board. (2013, February 18). *The Trouble with Online College*. Retrieved from The New York Times: http://www.nytimes.com/2013/02/19/opinion/the-trouble-with-online-college.html?ref=todayspaper&_r=3&

Tinto, V., & Pusser, B. (206, June). Moving From Theory to Action: Building a Model of Institutional Action for Student Success. Retrieved from http://nces.ed.gov/npec/pdf/Tinto_Pusser_Report.pdf

Tyler-Smith, & Keith. (n.d.). *Early Attrition among First Time eLearners: A Review of Factors that Contribute to Drop-out, Withdrawal and Non-completion Rates of Adult Learners undertaking eLearning Programmes*. Retrieved from Journal of Online Learning and Teaching: http://jolt.merlot.org/vol2no2/tyler-smith.htm

W., M. G. (2005). Challenge and Changes to Tinto's Persistence Theory: A Historical Review. *Journal of College Student Retention: Research, Theory and Practice, 6*, 191-207.

Wilcoxson, L., Cotter, J., & Joy, S. (2011). Beyond the First-Year Experience: the Impact on Attrition of Student Experiences throughout Undergraduate Degree Studies in Six Diverse Universities. *Studies in Higher Education, 36*, 331-352.

Appendix – Set Up HelperBot

1. Create a Bot on Azure, version 3, following these instructions: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-quickstart?view=azure-bot-service-3.0
2. Connect the bot to Slack, see: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-channel-connect-slack?view=azure-bot-service-3.0
3. Download the zip file containing the source code and the SQL creation script.
4. Create the database on Azure, using the SQL script: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-portal
5. In Visual Studio 2017 or higher, restore NuGet packages, configure the web.config settings, in particular:
    a. your Microsoft App ID/Password, see https://blog.botframework.com/2018/07/03/find-your-azure-bots-appid-and-appsecret/
    b. Create a password for the admin dialogue tree and set in the PASSW value
    c. Update the database config strings, one for Hangfire and the other for HelperBot
    d. Run the add me dialog, to get the bot ID saved in the messages table in the database
    e. Add the bot ID value in the web.config
    f. Regenerate your EDMX file under Models.
6. Download and install Microsoft Bot Emulator to test and debug, https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-3.0
7. To run the database import tool, you will need to remove any excess double quotes in the headers of the CSV file. Also, if there are any new or removed headers the code will have to updated in the console import tool:
    a. Add/remove in CSVFileDefinition
    b. Add/remove in `CSVFileDefinitionMap()`